Scalable Data Analysis

"I have had my results for a long time, but I do not yet know how I am to arrive at them."

–Carl Friedrich Gauss, 1777-1855

Jet data courtesy Kwan-Liu Ma, UC Davis. Image courtesy Wes Kendall, UTK

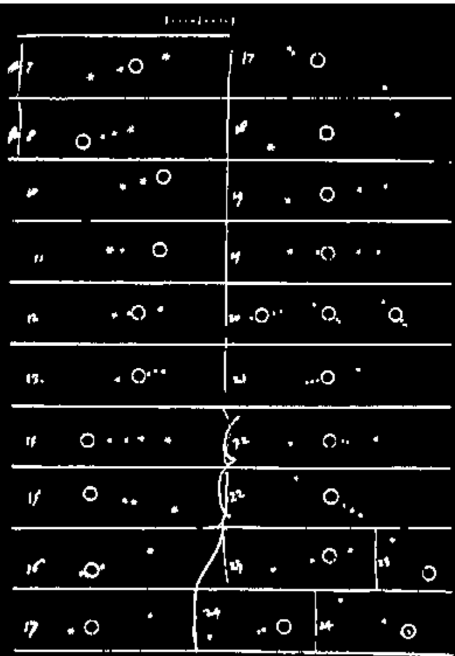
Tom Peterka

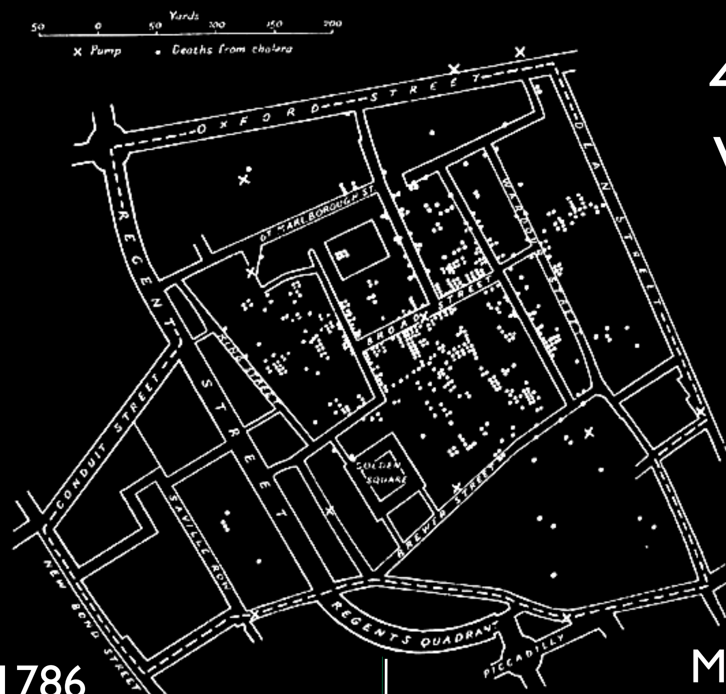
tpeterka@mcs.anl.gov

Mathematics and Computer Science Division

LANS Seminar 6/9/10

400 Years of Visualization



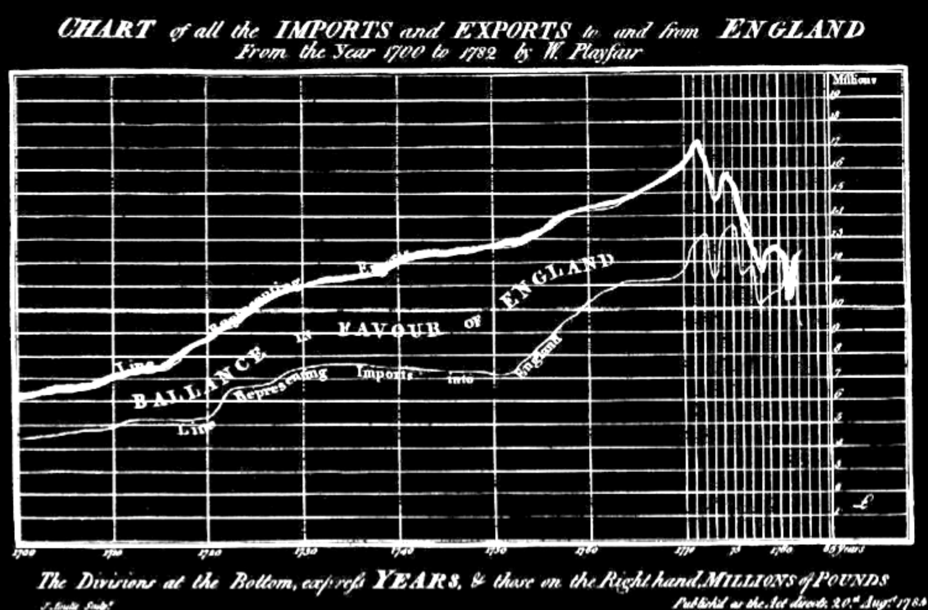


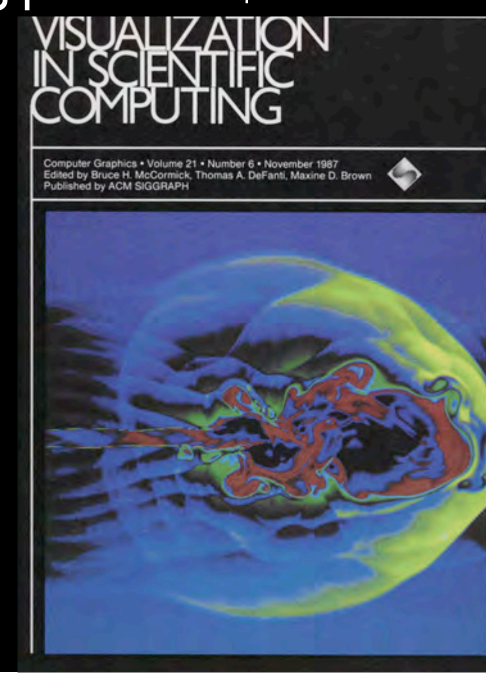
William Playfair, 1786

Galileo, 1610

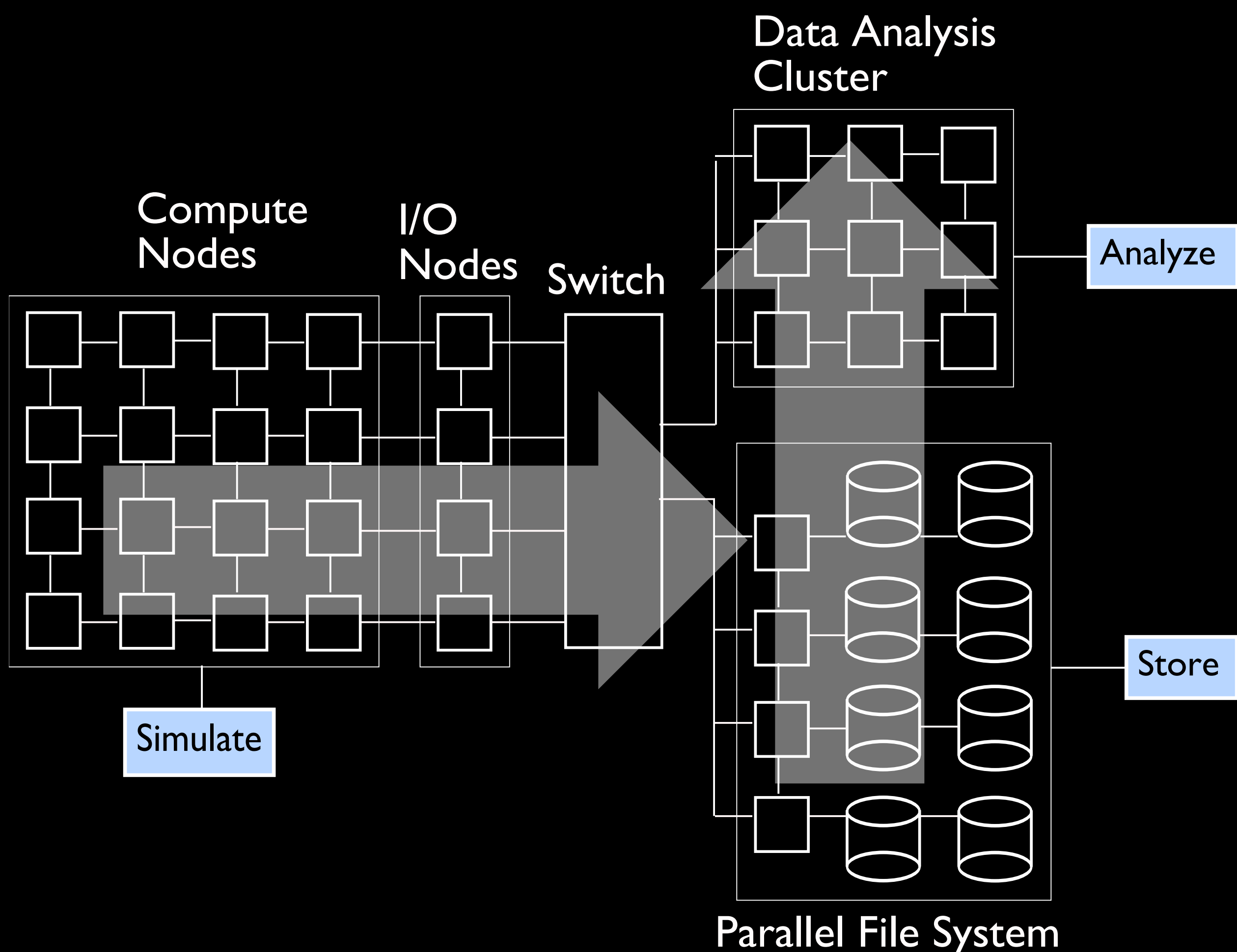
John Snow, 1854

McCormick et al., 1987





Scientific Data Analysis in HPC Environments



A linear, sequential pipeline where tasks mapped to architectures in fixed fashion is robust but not necessarily scalable.

"Models … produce data in amounts that make storage expensive, movement cumbersome, visualization difficult, and detailed analysis impossible. The result is a significantly reduced scientific return from the nation's largest computational efforts." -Mark Rast, Laboratory for Atmospheric and Space Physics, University of Colorado

The Data-Intensive Nature of Computing and Analysis

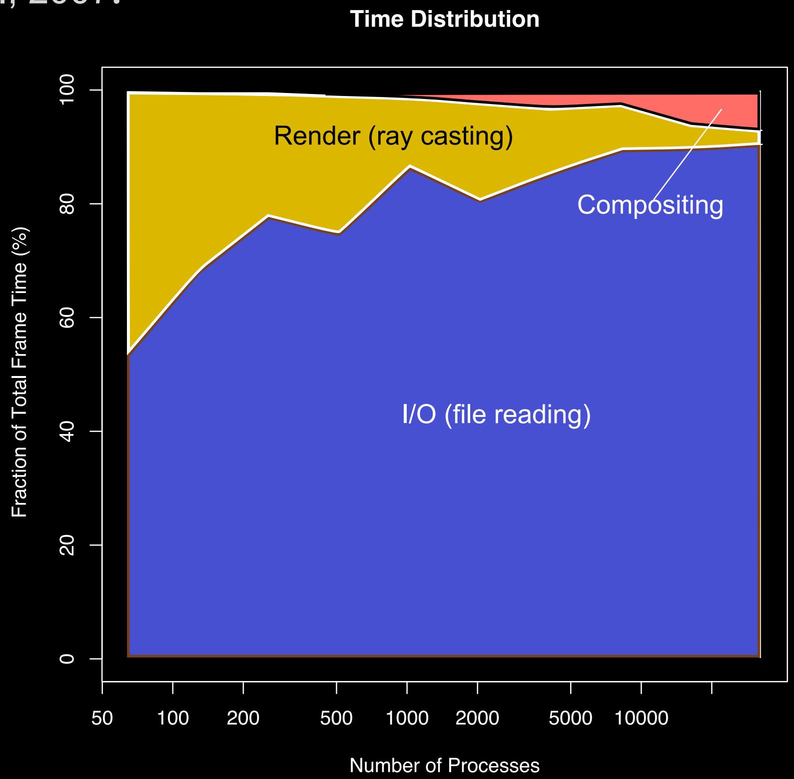
"Datasets being produced by experiments and simulations are rapidly outstripping our ability to explore and understand them" –Johnson et al., 2007.

Normalized Storage / Compute Metrics

Machine	FLOPS (Pflop/s)	Storage B/W (GB/s)	Flops per byte stored	Bytes comp. per byte stored
LLNL BG/L	0.6	43	$O(10^4)$	$O(10^3)$
Jaguar XT4	0.3	42	$O(10^4)$	$O(10^3)$
Intrepid BG/P	0.6	50	$O(10^4)$	$O(10^3)$
Roadrunner	1.0	50	$O(10^5)$	$O(10^4)$
Jaguar XT5	1.4	42	$O(10^5)$	$O(10^4)$

-In 2001, Flops per bytes stored was approximately 500. Ref: John May, 2001.

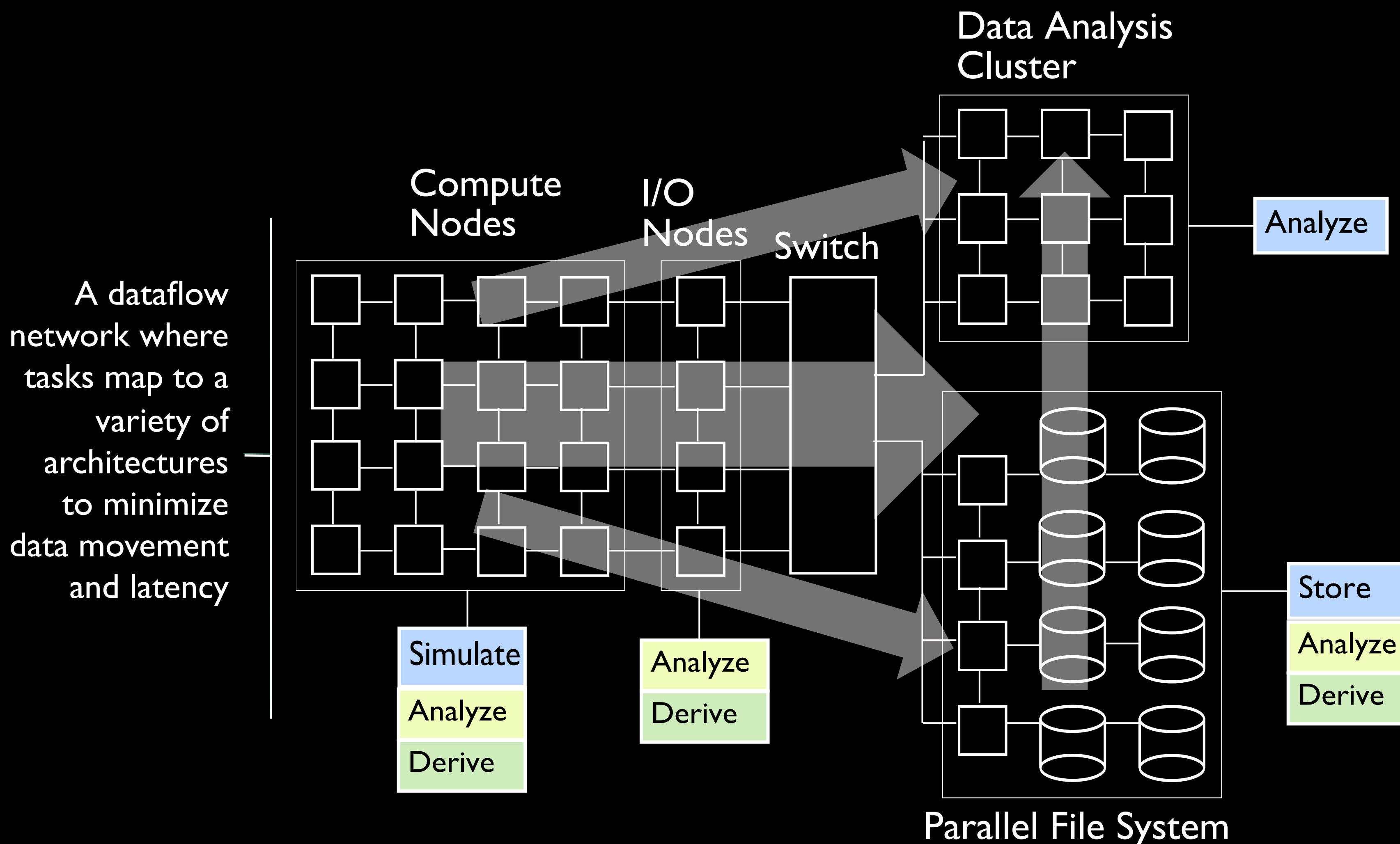
-DOE science applications generate results at an average rate of 40 flops per byte of data. Ref: Murphy et al. ICS'05.



The relative percentage of time in the stages of volume rendering as a function of system size. Large visualization is dominated by data movement: I/O and communication.

"Analysis and visualization will be limiting factors in gaining insight from exascale data."
–Dongarra et al., International Exascale Software Project Draft Road Map, 2009.

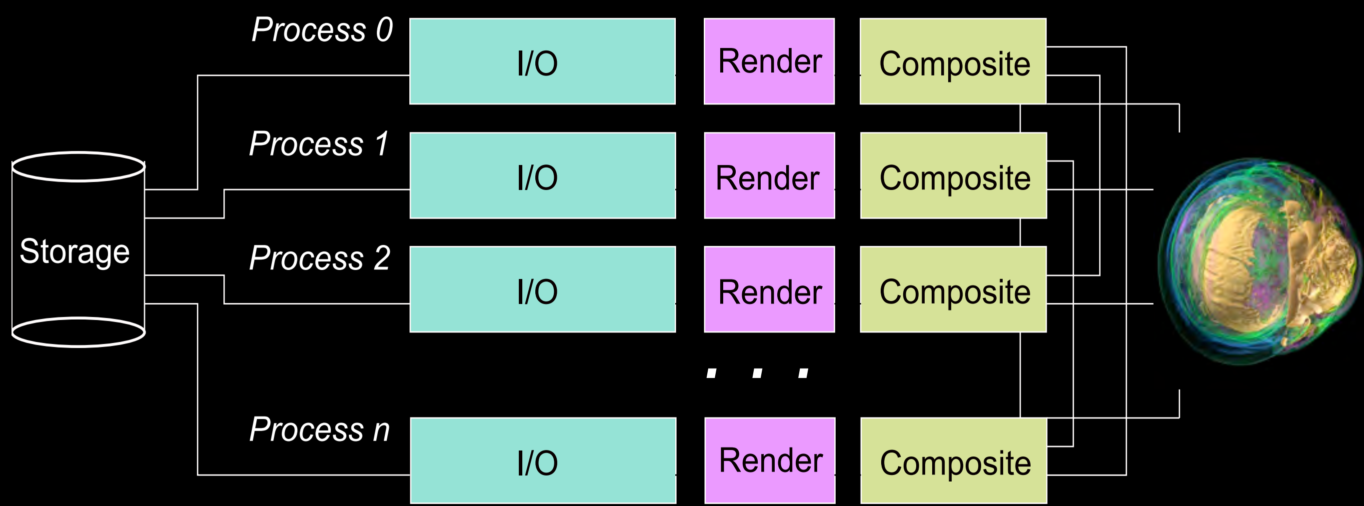
Scalable Data Analysis



"The combination of massive scale and complexity is such that high performance computers will be needed to analyze data, as well as to generate it through modeling and simulation."

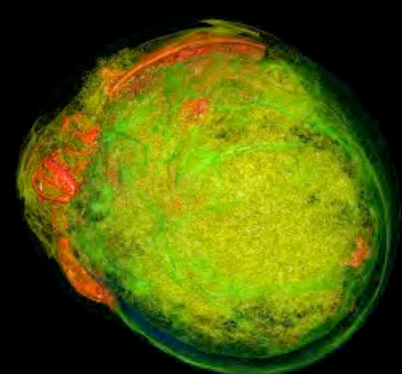
–Lucy Nowell, Scientific Data Management and Analysis at Extreme Scale, Office of Science Program Announcement LAB 10-256, 2010.

Large-Scale Parallel Volume Rendering



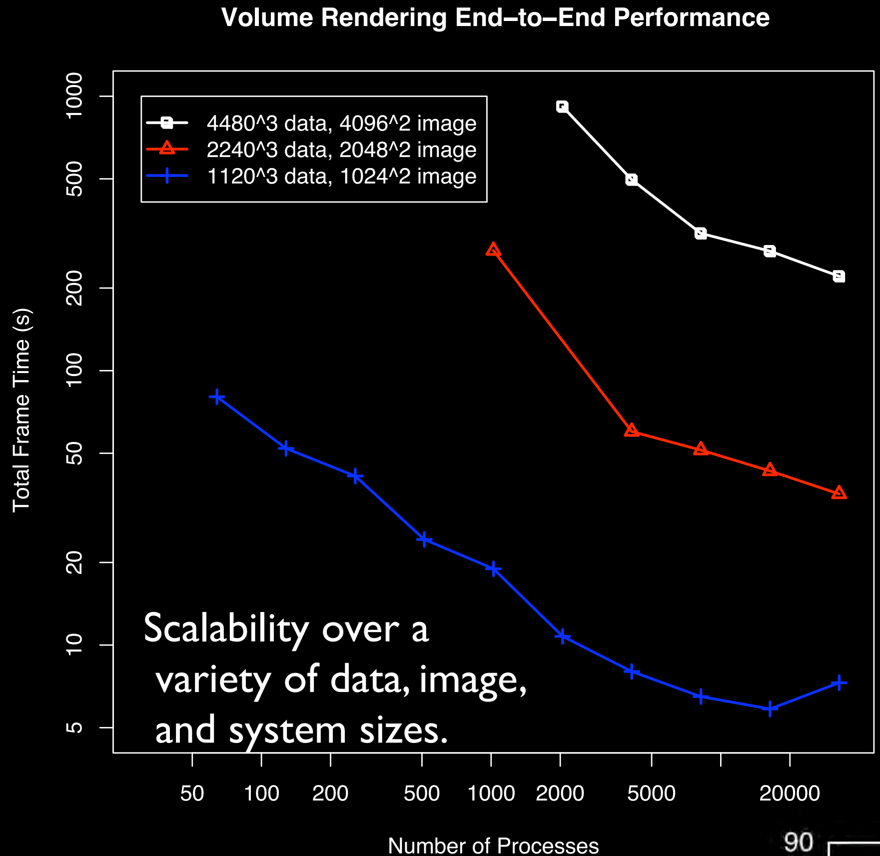
Parallel structure for volume rendering algorithm consists of 3 stages performed in parallel

Entropy over 100 time-steps



Parallel Volume Rendering on the IBM Blue Gene/P. EGPGV'08.

Benchmarking Performance

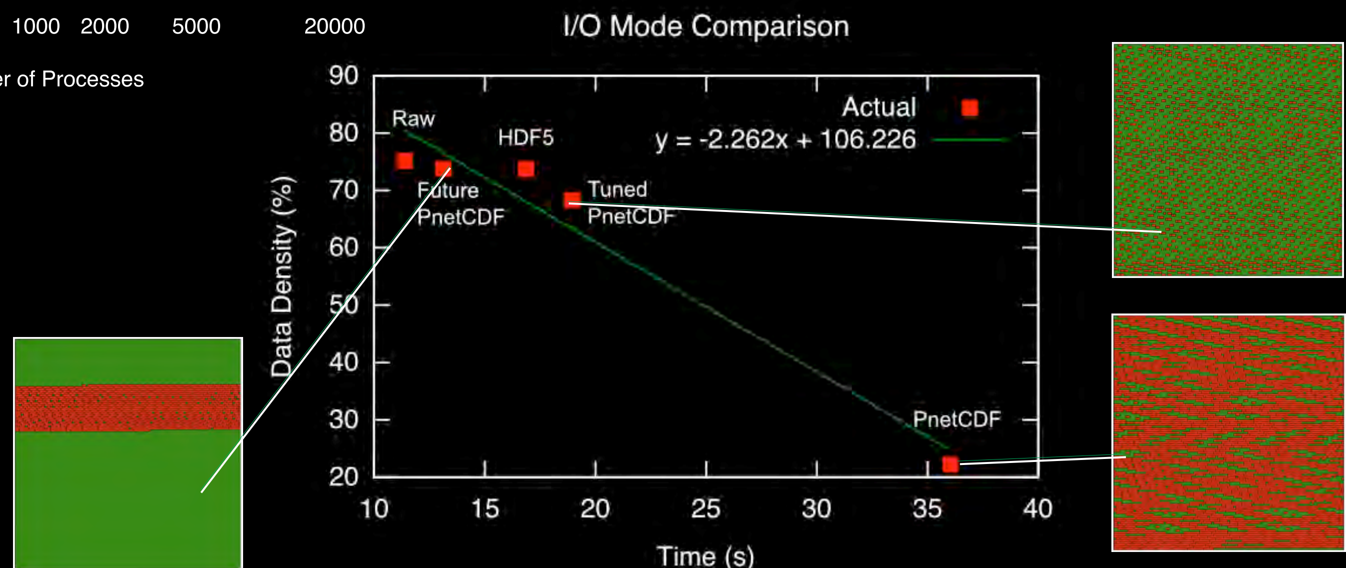


Scalability over a variety of data, image, and system sizes.

Grid Size	Time-step size (GB)	Image size (px)	# Procs	Tot. time (s)	% I/O	Read B/W (GB/s)
2240^3	42	2048^3	8K	51	96	0.9
			16K	43	97	1.0
			32K	35	96	1.3
4480^3	335	4096^3	8K	316	96	1.1
			16K	272	97	1.3
			32K	220	96	1.6

Volume rendering performance at large size is dominated by I/O.

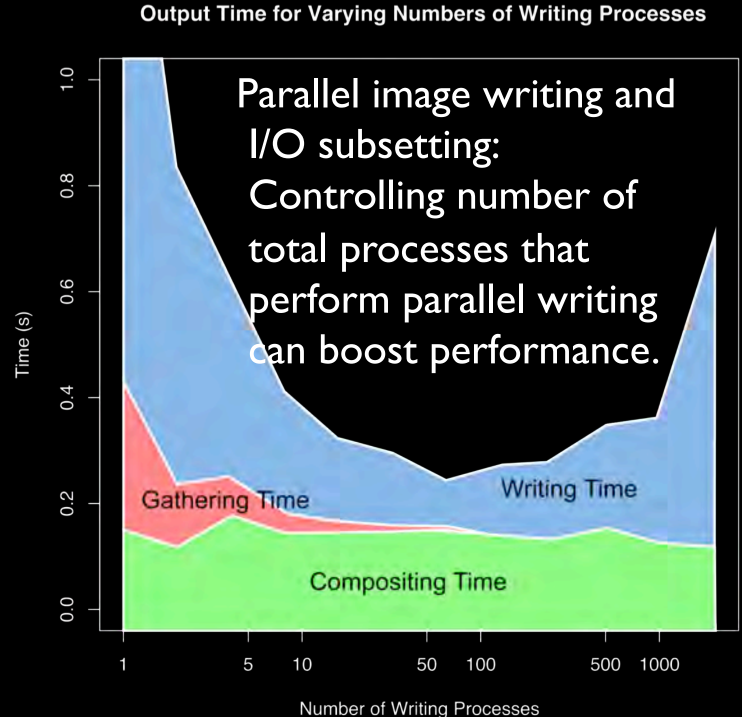
Changing data file layout can improve I/O performance, shown by access pattern signatures and performance data.



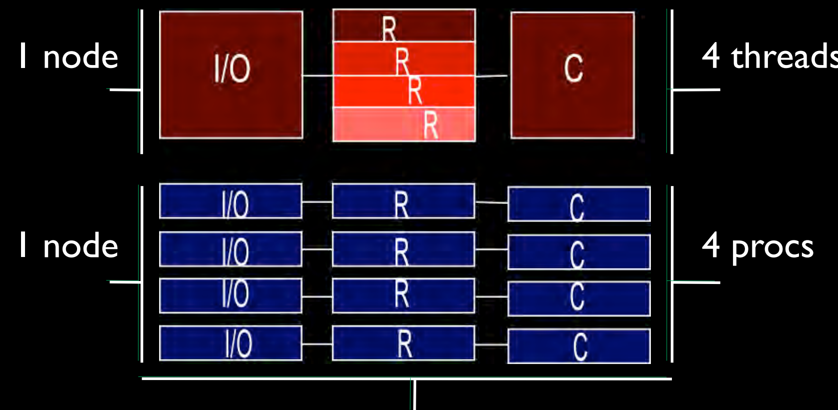
End-to-End Study of Parallel Volume Rendering on the IBM Blue Gene/P. ICPP'09.

Assessing Improvements to the Parallel Volume Rendering Pipeline at Large Scale. SC08 Ultrascale Visualization Workshop.

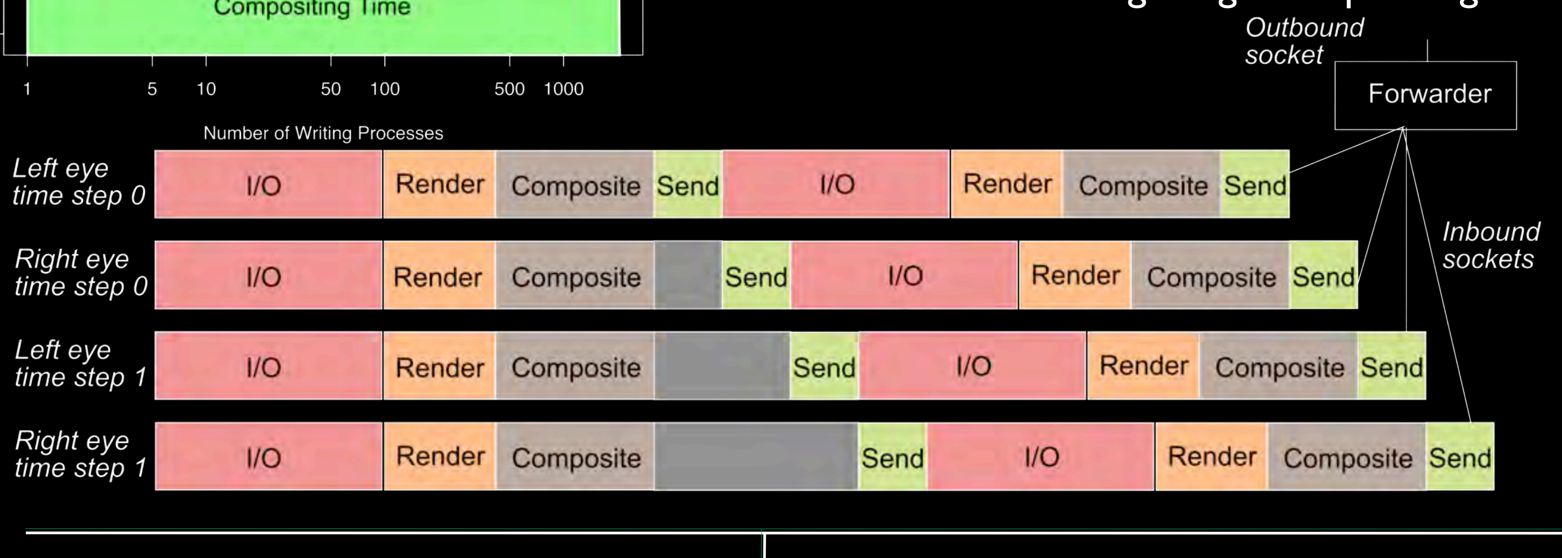
I/O Aggregation and Hybrid Parallelism



Parallel image writing and I/O subsetting: Controlling number of total processes that perform parallel writing can boost performance.



Combining threads with processes reduces interprocess communication during image compositing.



Parallel pipelining: I/O latency in a time series can be masked by visualizing multiple time steps in parallel pipelines. Each of the pipelines below is further parallelized among multiple nodes.

Large Scale Parallel Image Compositing

The final stage in sort-last parallel visualization algorithms:

1. Partition data among processes
2. Visualize local data
3. Composite resulting images into one

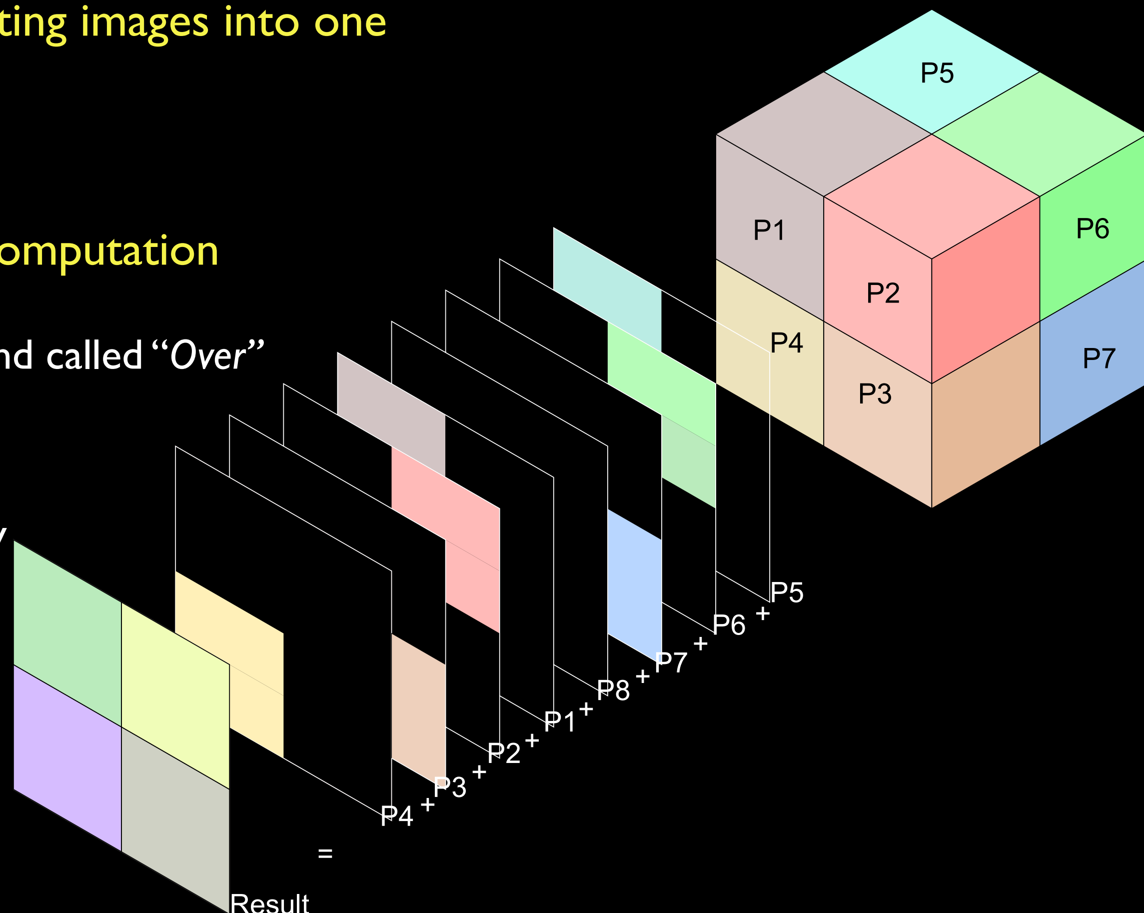
Composition = communication + computation

The computation is usually an alpha-blend called *"Over"*

$$i = (1.0 - \alpha_{old}) * i_{new} + i_{old}$$

$$\alpha = (1.0 - \alpha_{old}) * \alpha_{new} + \alpha_{old}$$

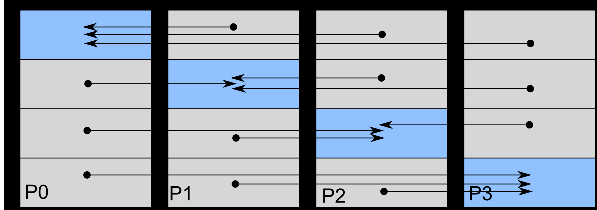
where i = intensity (R,G,B), α = opacity



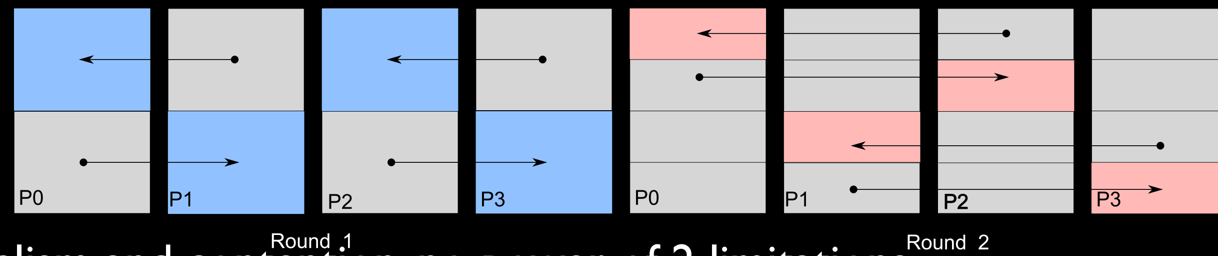
A Configurable Algorithm for Parallel Image-Compositing Applications. Peterka et al., SC09

Direct-Send, Binary Swap, and Radix-k

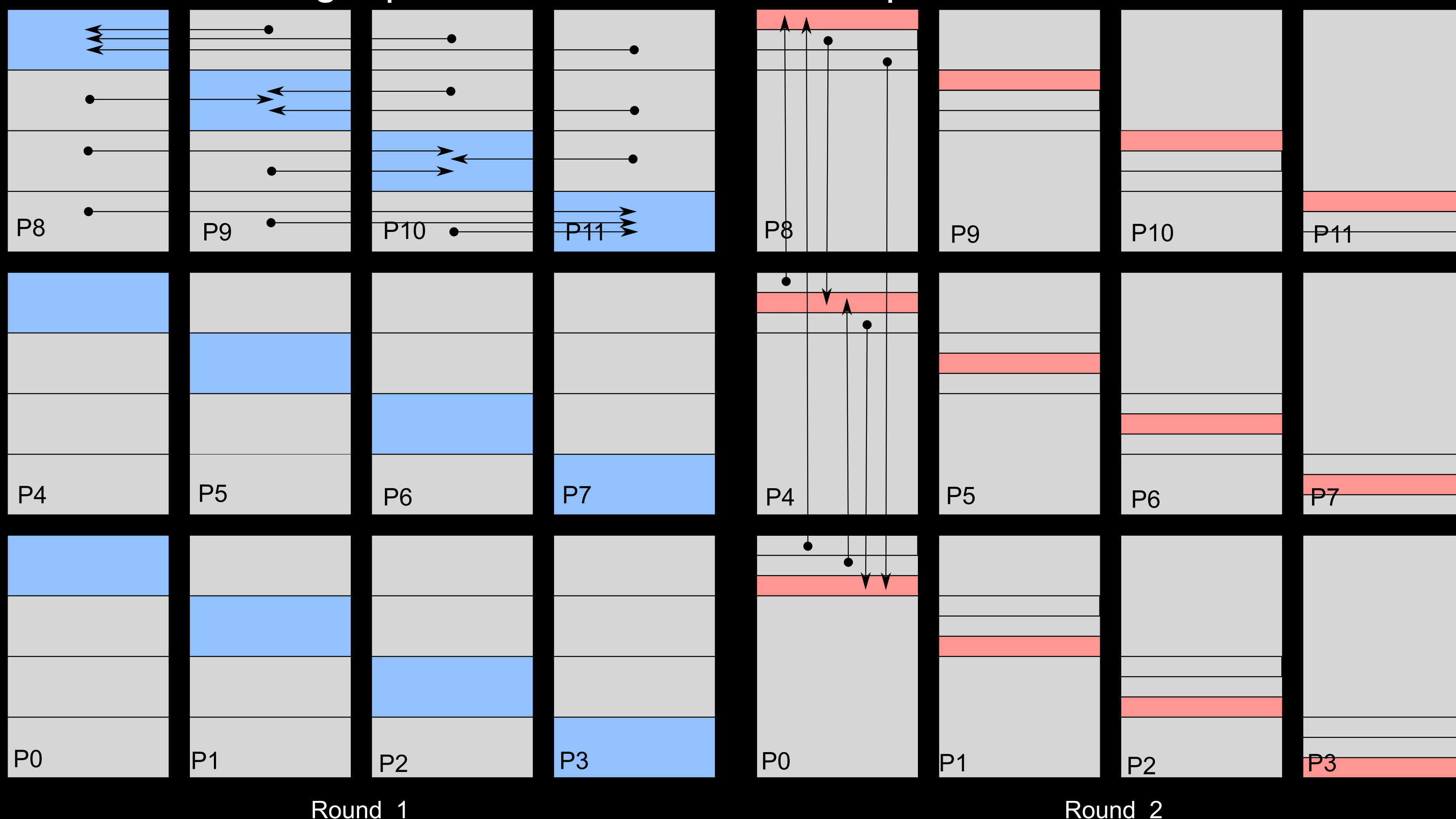
Direct-send: Parallel, contentious



Binary swap: Low parallelism, limited to powers of 2



Radix-k: Managed parallelism and contention, no power of 2 limitations



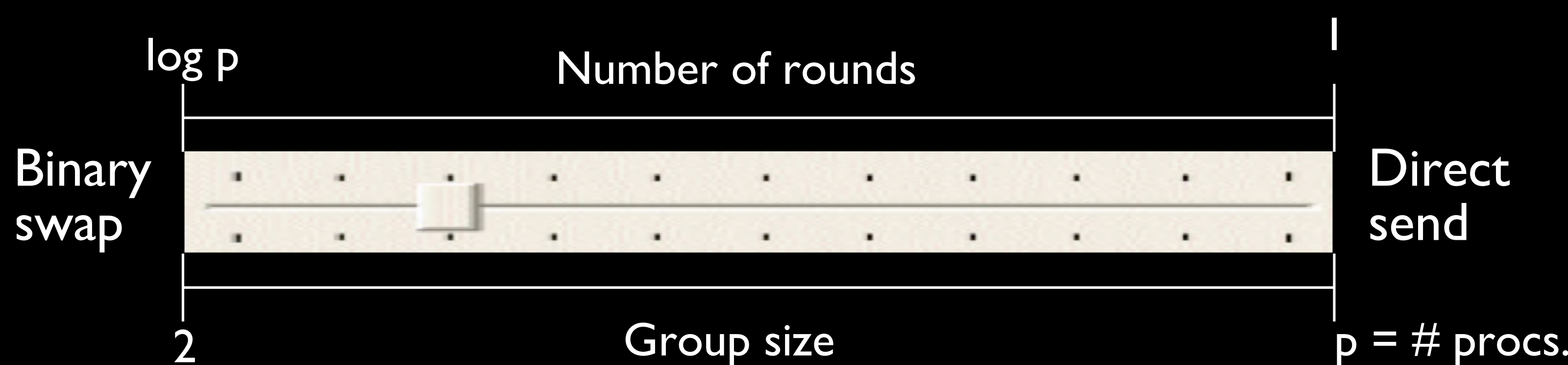
Radix-k: Configurable to Different Architectures

-Increase Concurrency: More participants per group than binary swap (k > 2)

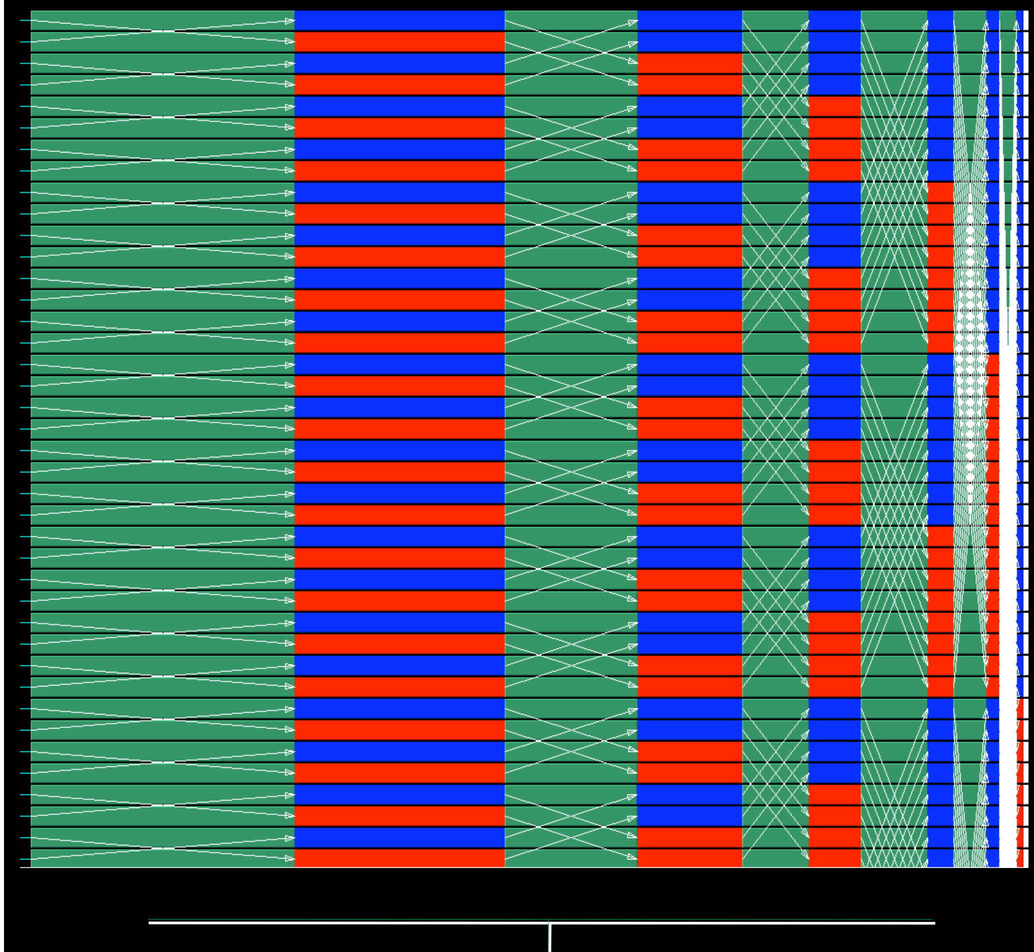
-Manage contention: limiting k value (k < p)

-Overlap communication with computation: nonblocking and careful ordering of operation

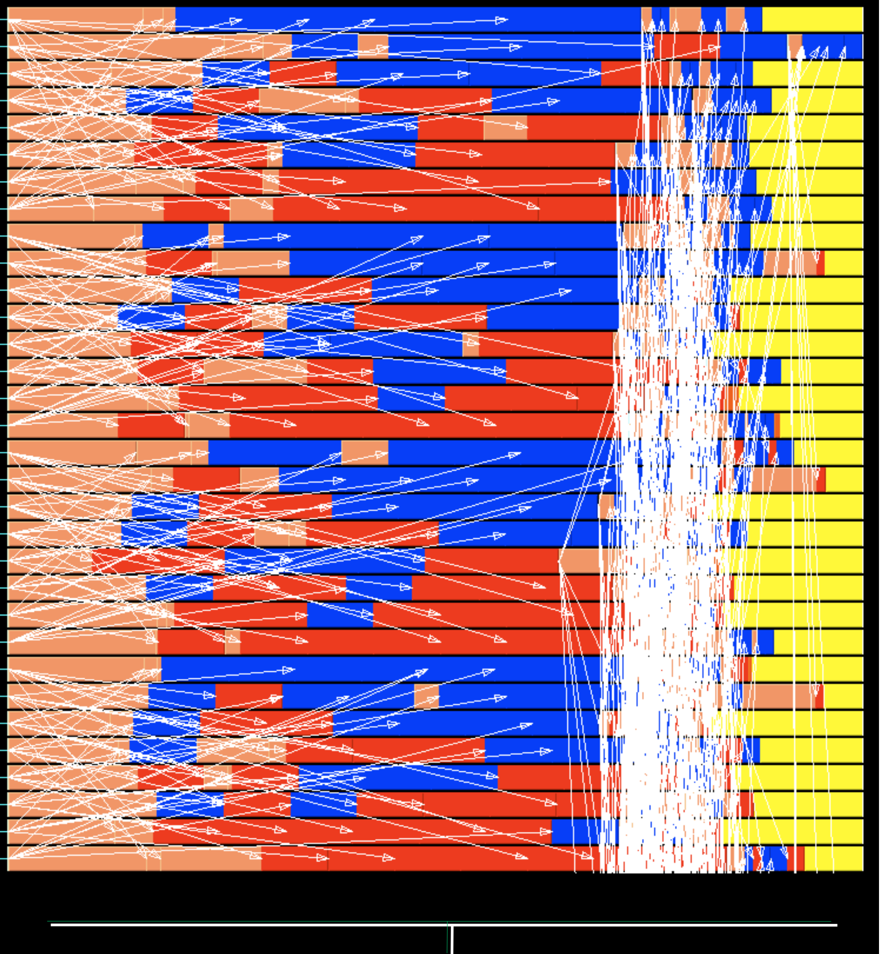
-No penalty for non-powers-of two numbers of processes: inherent in the algorithm design



Profiling Actual Cost with MPE & Jumpshot

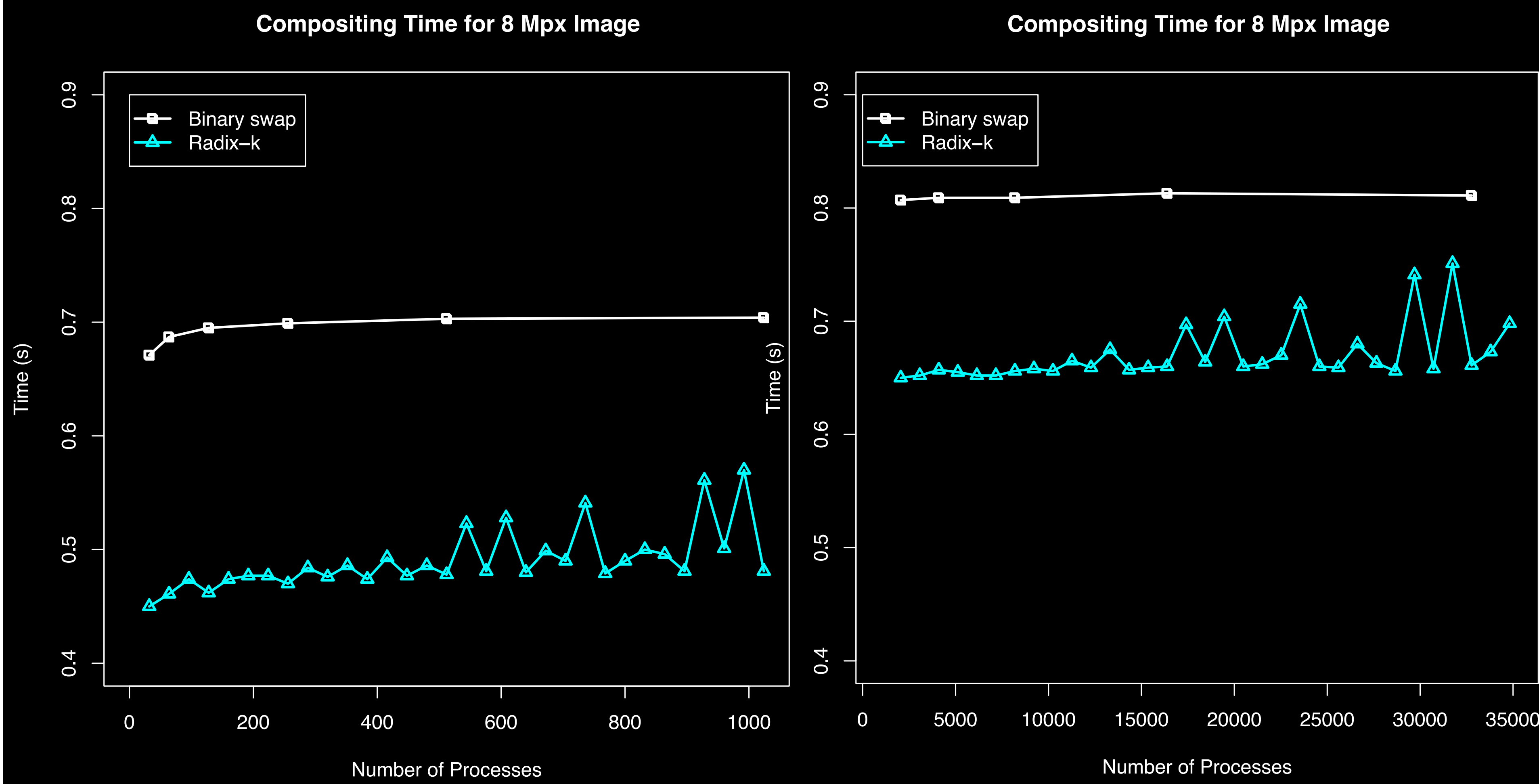


Jumpshot profile of binary swap for 64 processes is highly synchronized into 6 compute – communication rounds.



Radix-k for 64 processes factored into 2 rounds of k = [8, 8] overlaps communication with computation whenever possible.

Radix-k Performance on Blue Gene/P Intrepid



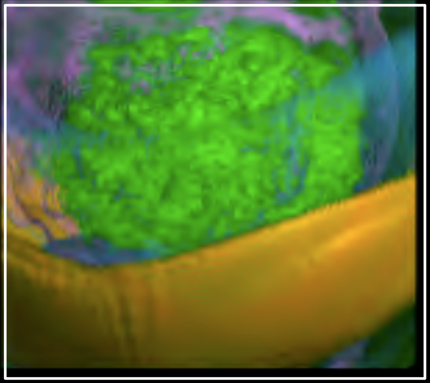
Radix-k improves 40% over binary swap at non-powers-of-two process counts. Left: p varies from 32 to 1024 in steps of 32. Right: p continues from 1024 to 35,000 in steps of 1024.

Optimized Radix-k at Scale

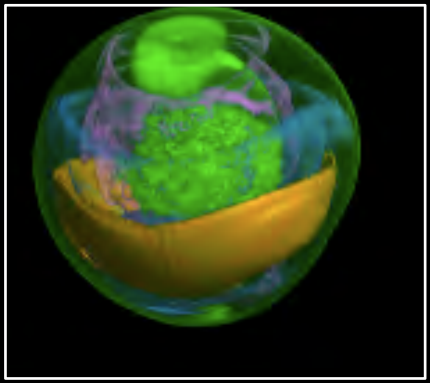
Benchmarked target k-values for Intrepid and other machines after RLE and bounding box optimizations.

P	4 Mpix	8 Mpix	16 Mpix	32 Mpix
8	8	8	8	8
16	16	16	16	16
32	32	32	32	32
64	64	64	64	64
128	64	128	128	128
256	64	128	128	128
512	64	128	128	128
1 K	64	32	128	128
2 K	32	32	128	128
4 K	32	32	32	32
8 K	32	32	32	32
16 K	32	32	32	32
32 K	32	32	32	32

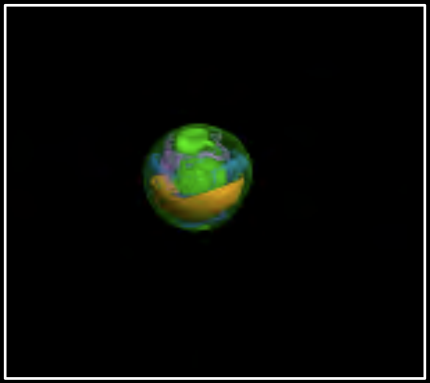
zoom = 3.0



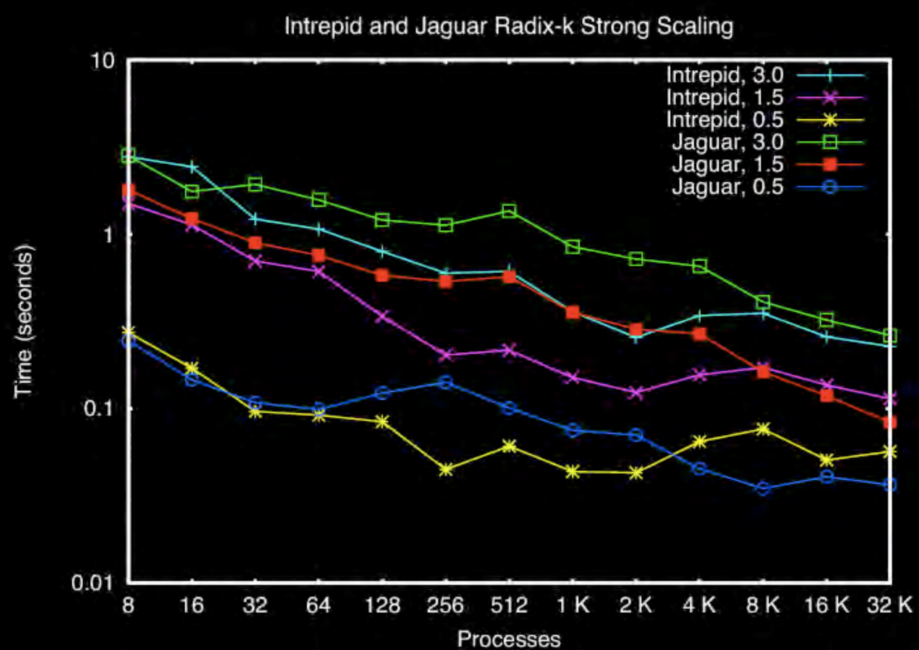
zoom = 1.5



zoom = 0.5

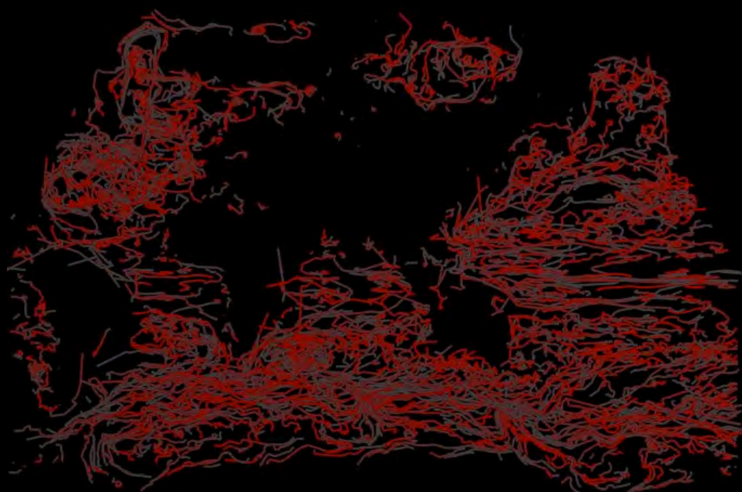


Examples of volume rendering at the 3 zoom levels shown below



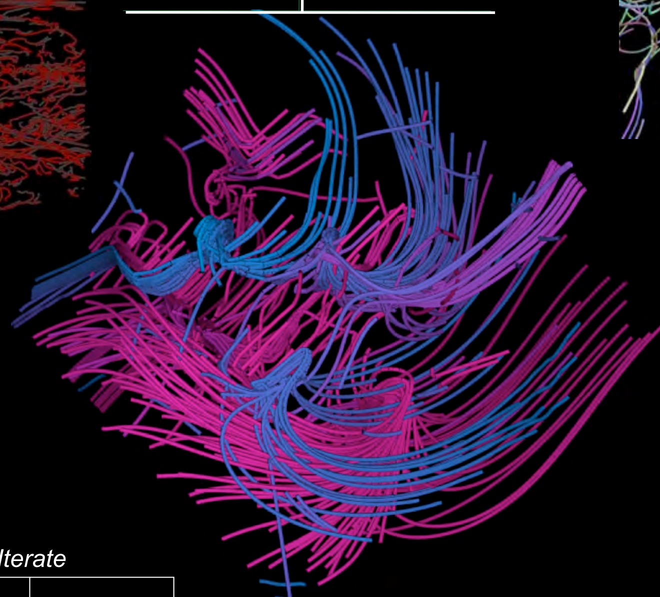
3X – 6X improvement over optimized binary swap (with bounding boxes and RLE) in many cases. 64Mpix at 32K processes can be composited at .08 s, or 12.5 fps.

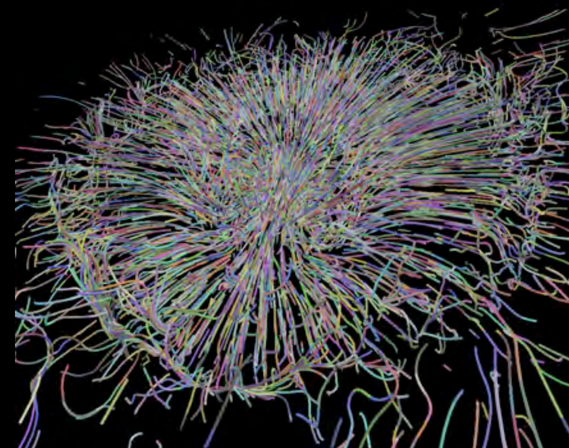
Large-Scale Parallel Particle Tracing



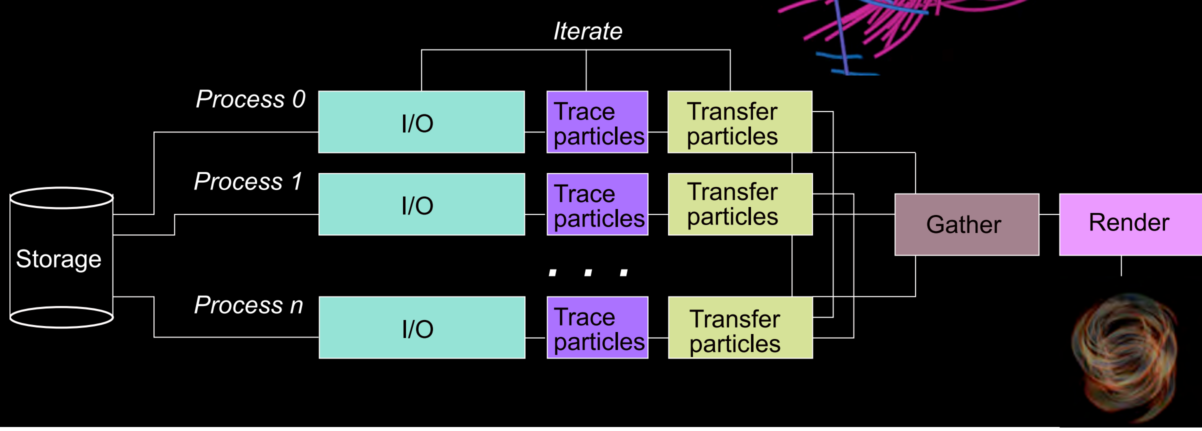
Ocean current data courtesy Rob Jacob, ANL

Jet data courtesy Kwan-Liu Ma, UC Davis



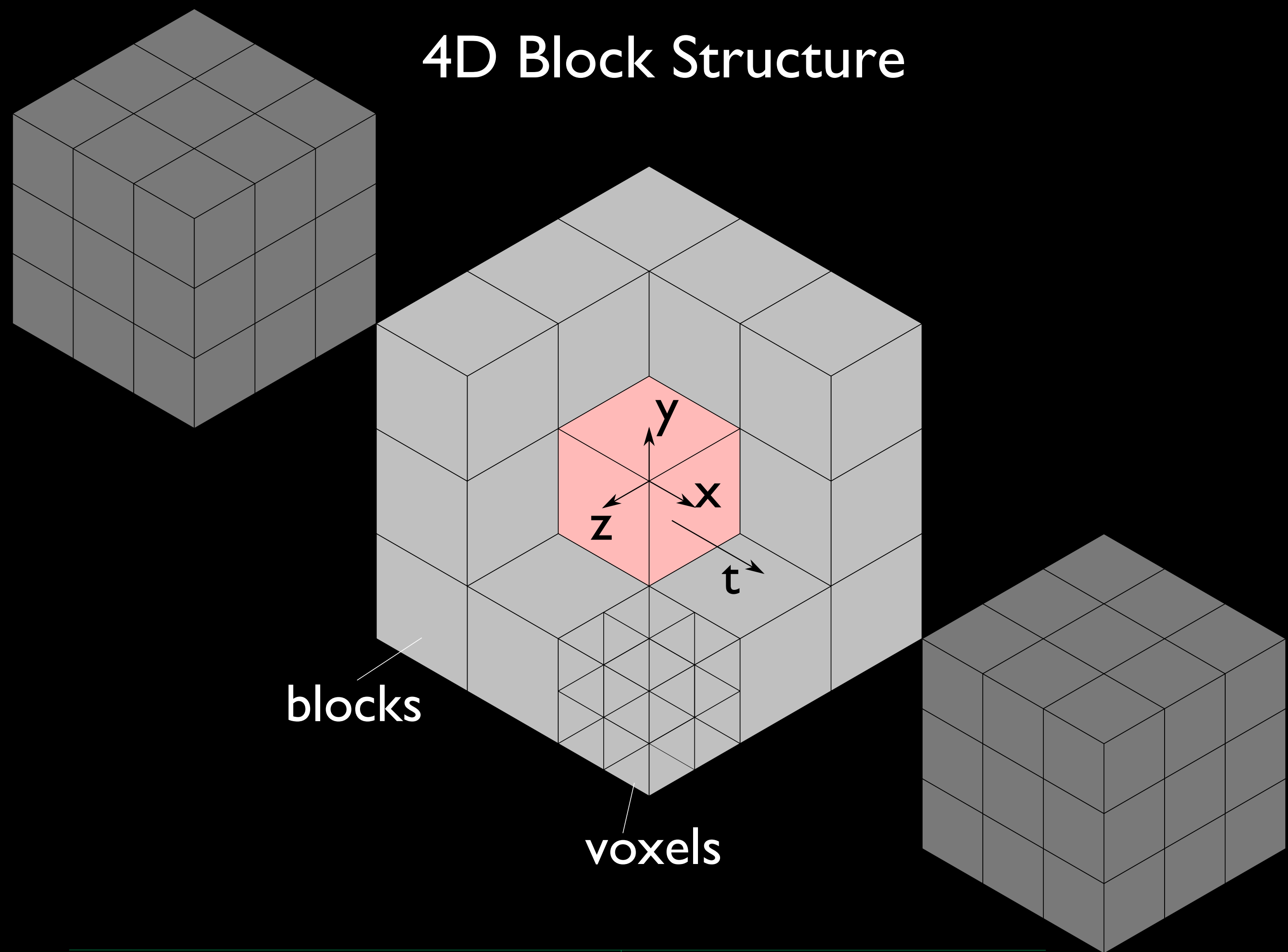


Type IA supernova data courtesy George Jordan, UofC FLASH Center



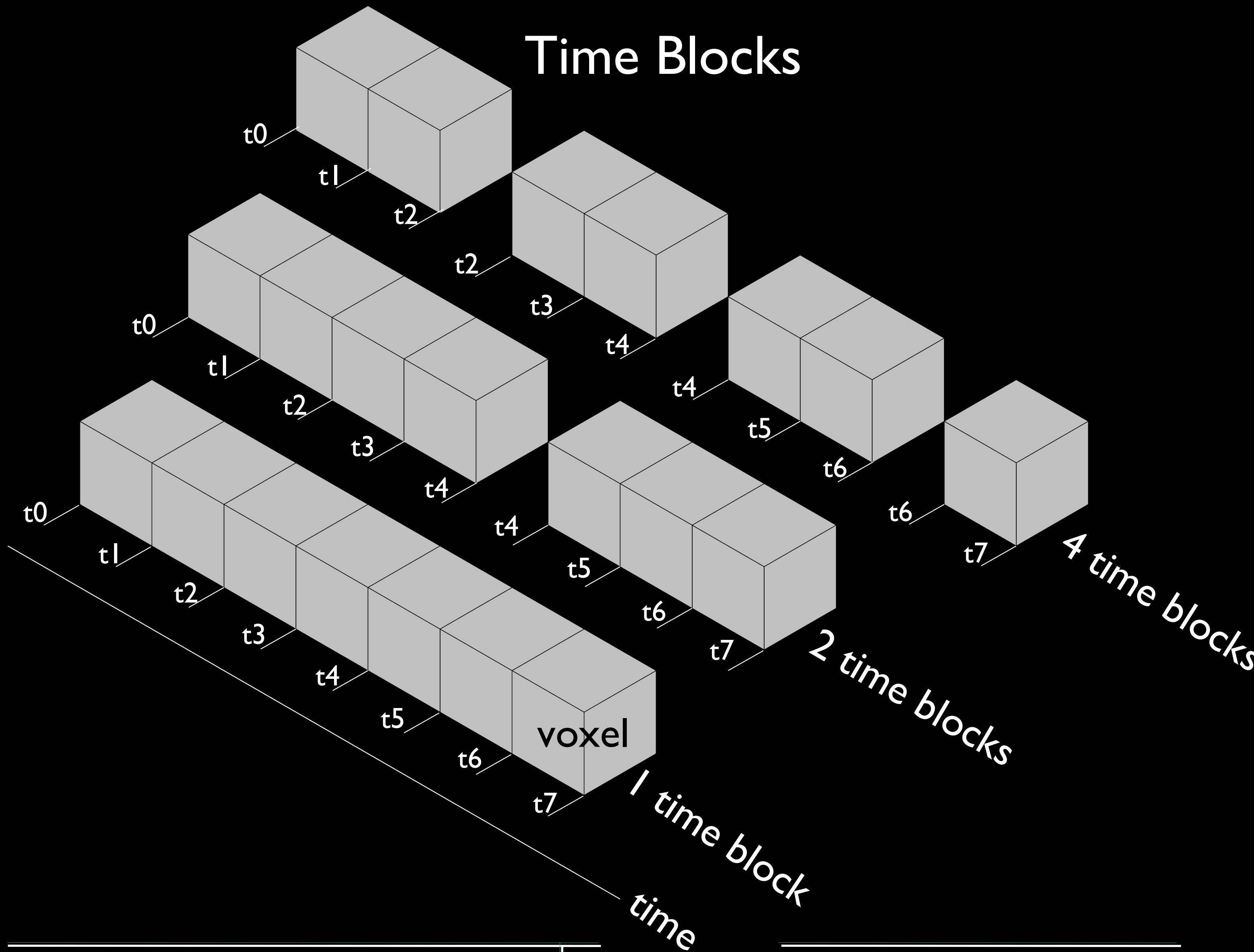
Parallel structure for flow visualization algorithm consists of iterations of particle tracing and transfer, followed by a rendering stage.

4D Block Structure



- -True 4D blocks
- -Blocks consist of 4D voxels (eg 16x16x16x4 time steps)
- -Messages are sent when any of the 4 extents are exceeded
- -3^4 = 81 neighbors for regular grid, counting self
- -Variable number of neighbors for AMR grid

Time Blocks



-A way to control in-core / out-of-core behavior

-One time block resident in memory at any one time

-Memory distributed in spatial (x,y,z) dimensions, serialized in time dimension

ExchangeNeighbors()

Old

Organize sending block ids, # points, by process rank

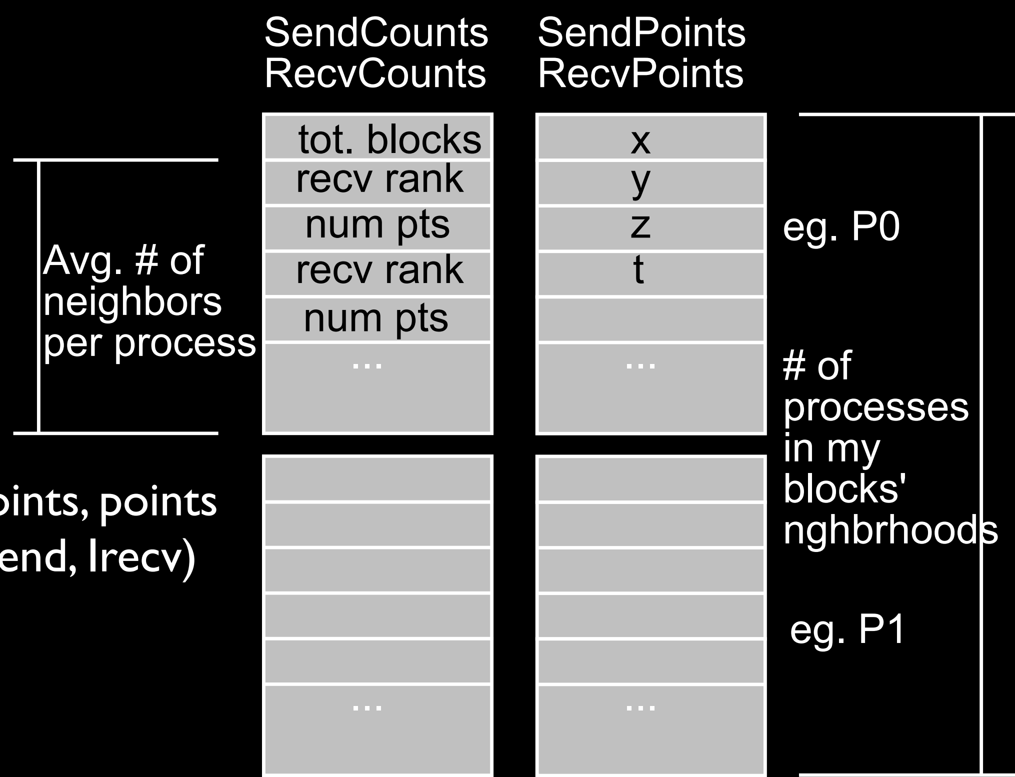
Exchange point counts (MPI_Allltoallv)

Unpack vector of receiving point counts

Pack vector of sending points

Exchange points (MPI_Alltoallv)

Unpack vector of received points



Current

For each neighbor,

  Pack messages of sending block ids, # points, points

Exchange point counts and points (MPI_Isend, Irecv)

For each neighbor,

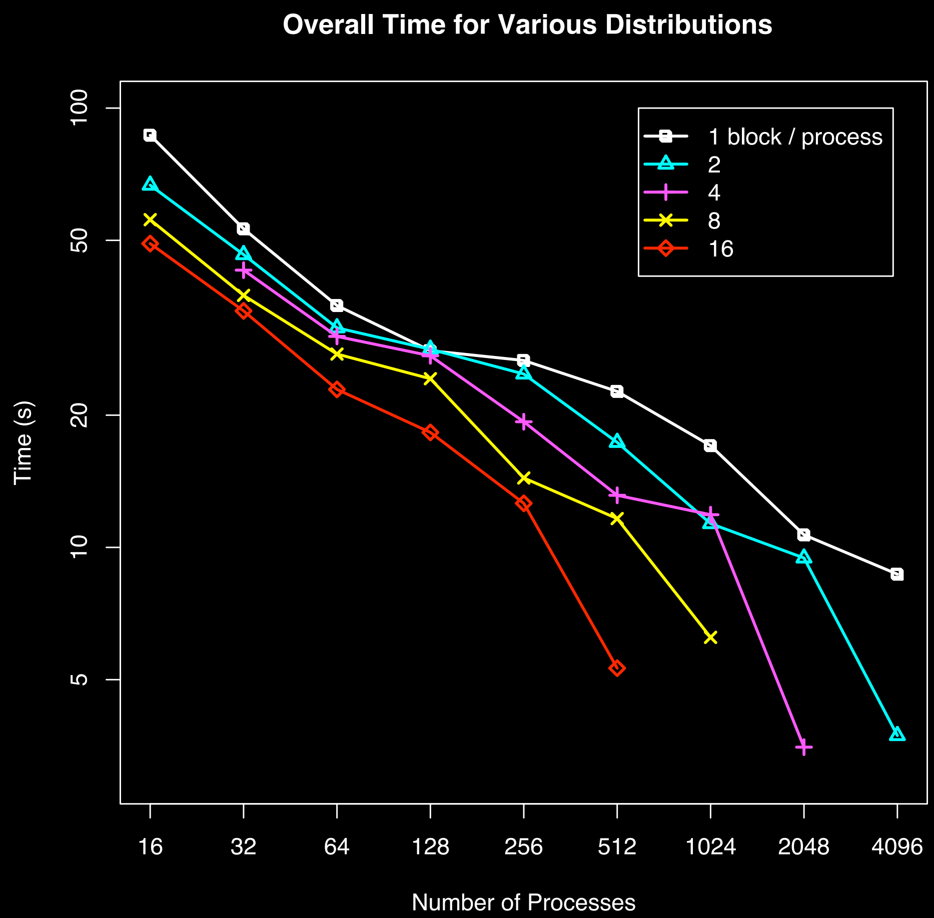
  Unpack vector of receiving points

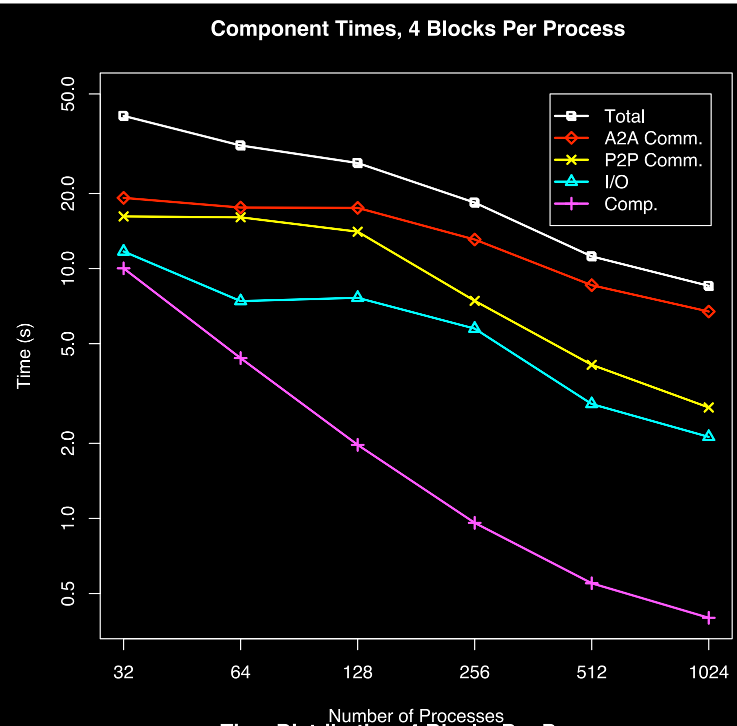
Future

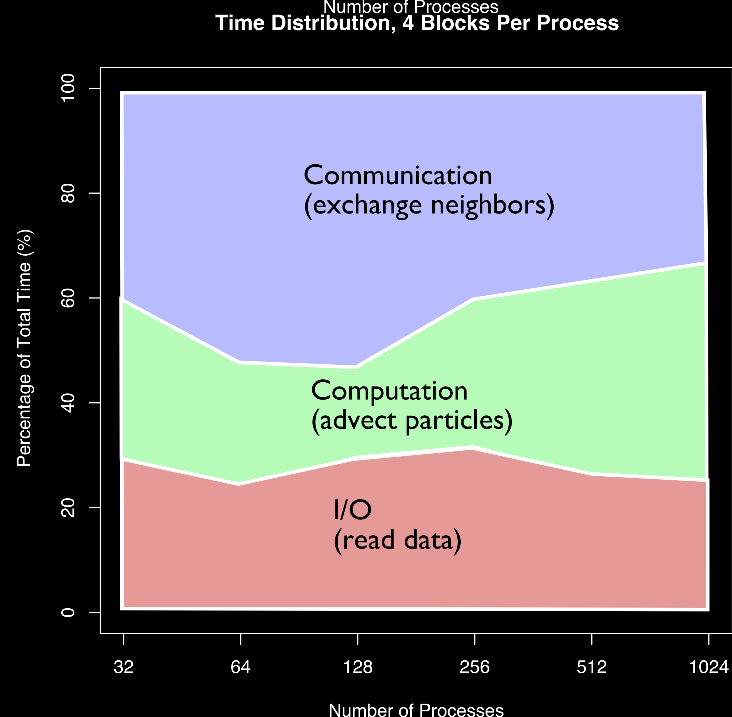
-MPI-2 one sided communication

-LibNBC sparse nonblocking collectives

Strong Scaling Baseline Performance

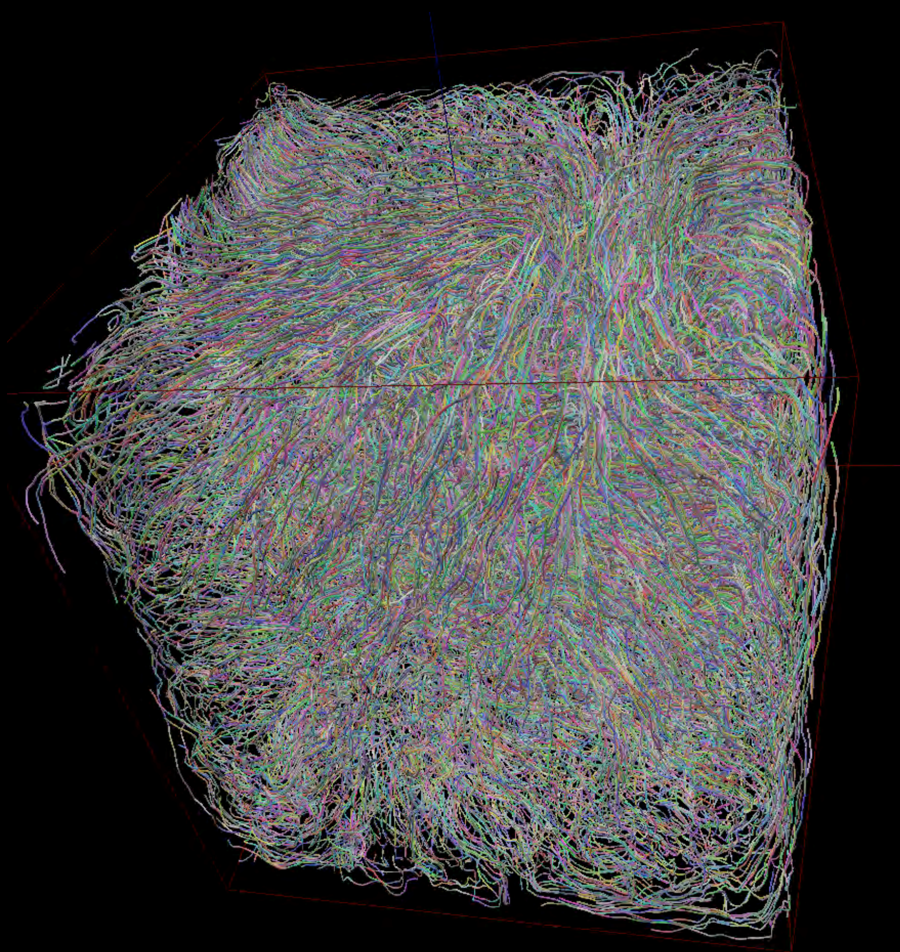


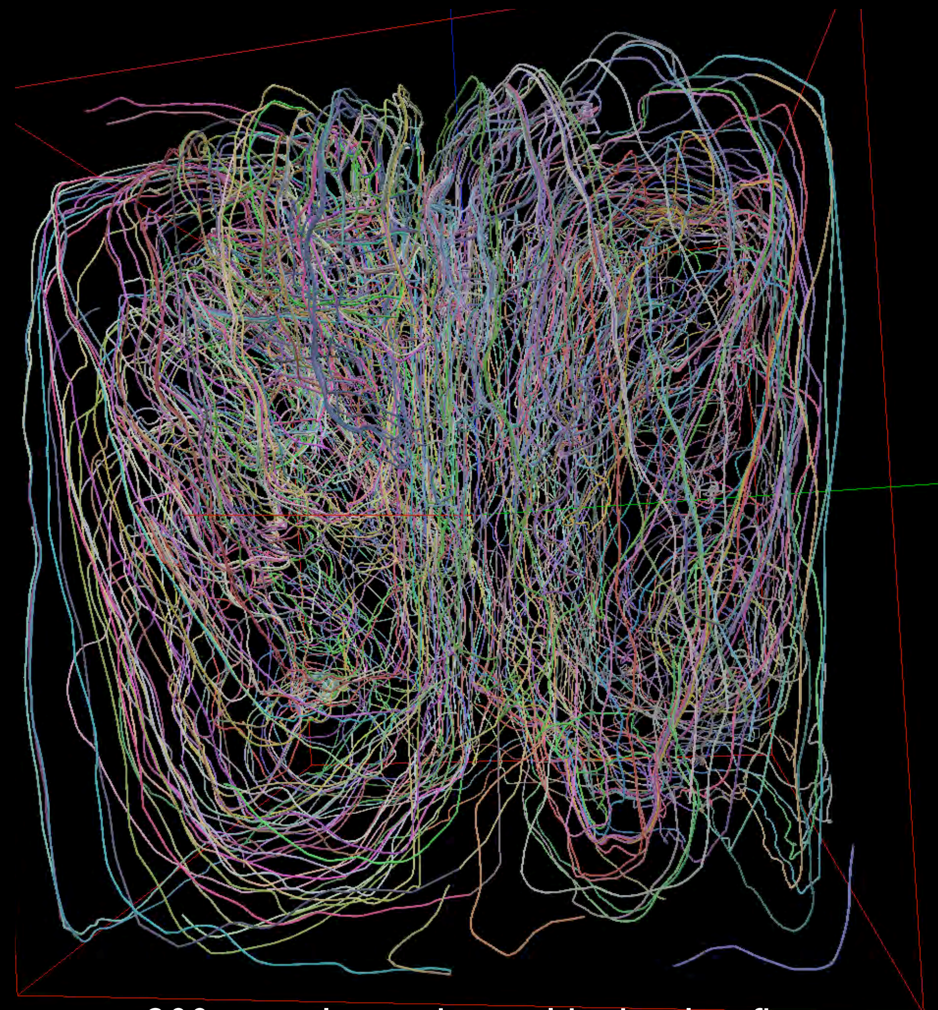




Thermal hydraulics flow. 134M cells, 8K particles.
1,2,4,8,16 round robin blocks per process.

Particle Density: What 8K Particles Look Like





8K particles in thermal hydraulics flow. Dense seeding, while not visually useful, is necessary for querying flow features and generating derived fields such as divergence using FTLE.

200 particles in thermal hydraulics flow. Sparse seeding is useful for interactive visual exploration. Vortices and convective currents are evident.

Apply I/O and MPI Expertise to Data-Intensive Analysis

Conclusions

-HPC resources can be harnessed for scalable analysis
-Scalable analysis is data-intensive: Moving data, transforming data, interacting with data
-Detailed study of data movement, both network and storage, is needed
-Results impact application tools as well as systems software libs

Ongoing, Future

-Continue to collaborate with others in developing infrastructure for scalable analysis in other HPC subsystems

- -ION analysis
- -Coupling storage and analysis

-Strengthen collaborations with scientists to integrate analysis with applications

- -In situ analysis
- -Information-theoretic analysis

-Continue to develop immersive interfaces and environments for science

- -Immersive environments for material interfaces
- -NG-CAVE environment for a variety of scientific and medical applications

Thank you

Acknowledgments:

Facilities
Argonne Leadership Computing Facility (ALCF)
Oak Ridge National Center for Computational Sciences (NCCS)

Funding
US DOE SciDAC UltraVis Institute

People
Rob Ross, Han-Wei Shen, Jian Huang, Wes Kendall, Rajeev Thakur, Dave Goodell, Kwan-Liu Ma, Hongfeng Yu

"The purpose of computing is insight, not numbers."
–Richard Hamming, 1962

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division